A Short Report of My Solution to the Assignment

Wei Qiu

May 24, 2014

1 Directory structure

I reorganized the files to make my solution easier to explain.

- All the data is located at data.
- Test data is in data/test. And test.en is copied to reference0.
- Train data is in data/train.
- Dev data is in data/dev.

2 Problem 1 & 3

The config file for the baseline system is located at baseline/config. The config file for the baseline system with further tuning on development set is located at baseline-tune/config.

The output is located at evaluation directory correspondingly. My first attempt of using mert for tuning(which I used for my previous project) crashed surprisingly. I turned to the moses-support mailing list for help.

3 Problem 2

The first idea comes to my mind is to simply append all of the terminology table to the training table. The sript split_term.py is used to split the terminology into seperate files. Then I manually append them to the corresponding training files.

The trained model using this idea is located at prob2. The corresponding training data is at data/train_term.

A further research shows that moses has an advanced feature that can let the user specify how to translate certain words by marking up them during preprocessing.[1] This may be the best approach to deal with external terminology dictionary. (Using place holder may be another choice, but not as convenient as xml-input feature provided by moses.)

A python script src/markup wrap the items occurred in the terminology into xml format which can be accepted by moses. Corresponding experiment is located at prob2-xmlmarkup.

[1] `http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc11`

Name	Tuning	BLEU	METEOR	GTM	TER
Baseline	none	0.2307	0.2523	0.1951	0.6258
Baseline	batch-mira	0.0055	0.1449	0.0334	4.9259
Baseline	pro	0.2436	0.2601	0.1942	0.6382
Baseline+append terminology	batch-mira	0.2442	0.2638	0.1928	0.6512
Baseline+append terminology	pro	0.2444	0.2604	0.1952	0.6395
Baseline+XML markup	pro	0.2436	0.26	0.1942	0.6384
Compound words segment	none				
Word pre-ordering	none				

Table 1: Result

4 Problem 5: further improvement

There are at least two ways to improve the translation quality:

- As German is an agglutinative language, splitting up the compound words can reduce the sparsity. We can already find a lot of OOV words in the output of baseline system.
- As the word order in German has much more freedom than English, a preprocessing step which always normalize German sentence to SVO can reduce the sparsity as well.

For the first idea, I found tools such as jwordsplitter[2] and compound-splitter[3]. For the second idea, I found that the idea has been explored since Xia and McCord (2004). Collins, Koehn, and Kuerov (2005) was on German-English language pair.

5 Result

The results are presented in Table 1. The tuning step using batch-mira seem to be quite unreliable due to the relatively small size of dev set. So we only use PRO(Pairwise Rank Optimization) for xmlmarkup. To integrate the terminology, both methods show similar performance.

References

Collins, Michael, Philipp Koehn, and Ivona Kuerov (2005). "Clause restructuring for statistical machine translation". In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 531–540.

Xia, Fei and Michael McCord (2004). "Improving a statistical MT system with automatically learned rewrite patterns". In: *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, p. 508.

[2] `http://www.danielnaber.de/jwordsplitter/index_en.html`

[3] `https://github.com/dweiss/compound-splitter`